View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by DSpace at VSB Technical University of Ostrava

VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra telekomunikační techniky

Traffic Shaping pod Linuxem

Traffic Shapping with Linux

2009 David Buček

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal

V Ostravě 4. května 2009 ……

Podpis

Poděkování

Rád bych tímto poděkoval vedoucímu mé bakalářské práce Ing. Pavlu Nevludovi za odborné vedení a cenné rady, které mi pomohly při řešení této práce.

Abstrakt

Bakalářská práce se zabývá problematikou traffic shapingu pod linuxem.

Práce je rozdělena na tři části. První část se zabývá teoretickým rozborem problematiky, základními principy shapovaní trafficu (tvarování toku dat).

Druhá část je tvořena návrhem pracoviště pro konfiguraci, zprovoznění a testování traffic shapingu v poslední verzi linuxové distribuci Debian Lenny. Poslední třetí části se zabývám samotnou prezentací. Práce obsahuje popisy, schémata strukturované sítě, ukázky, vysvětlení postupů a jednoduchých skriptů pro linux.

Prezentace je provedena formou webových stránek. Stánky jsou založeny na redakčním systému Joomla, krátké animace ve flashi jednoduše znázorňují samotné chování shapingu.

Klíčová slova

Linux, Debian, traffic shaping

Abstract

My Bachelors work deals with problems of traffic shaping under Linux.

This work is devided into three parts. The first part acts with teoretical analysis of questions, the basic principles of Shaping the traffic (formating of data flow)

Second part is about designing the workplace for configuration, operationing and testing of traffic shaping under last version of linux system distribution Debian Lenny. The last part is based on own presentation graphics.

The work contains legend, schematics of structured system, ilustrations, explication of procedures and basic mimeographeds for linux.

The presentation is realized via web site pages. The web site is based on content management system Mr. Joomla and short animations in flash simply represents the own behaviour of shaping.

Keywords

Linux, Debian, traffic shaping

Seznam použitých zkratek a symbolů

DSCP	Differentiated Services Code Point – typ služeb
IP	Internet Protocol – protokol používaný v počítačových sítích
NAT	Network Address Translation - překlad síťových adres
OS	Operační systém
PC	Personal computer – osobní počítač
QoS	Quality of Service – kvalita služeb
TOS	Type of Services - typ služeb

Obsah

1 Úvod

Dnešní společností, ale i současným světem hýbou informační technologie. K informacím se nejčastěji přistupuje přes sítě internetu a intranetu. Dnešní student si nedokáže svět bez internetu ani představit. Po příchodu do školy zkontroluje příchozí poštu, prodlouží zapůjčení skript z knihovny, při objednávání jídla v menze zjistí, že nemá dostatek peněz na účtu tak si bezhotovostní platbou vloží peníze na účet menzy pomocí kreditní karty a to bez zvednutí se ze židle.

Ze všech stran na nás křičí reklamy o rychlejším a rychlejším internetu. Rychlost stahování dat u této služby není jediným parametrem, který by nás měl zajímat.

Cílem této bakalářské práce je popsat jak zprovoznit traffic shaping na volně šiřitelném operačním systému GNU Linux, v distribuci Debian i386. Výhodou tohoto použití je naprostá bezplatnost, tímto se sníží náklady na vybudování a na provoz.

První teoretická část se zabývá teoretickým rozborem problematiky, základními principy shapovaní trafficu a zařízení na kterých lze tuto funkci využít.

Druhá část je tvořena praktickým návrhem laboratorního pracoviště pro konfiguraci, zprovoznění a testování traffic shapingu v poslední verzi linuxové distribuci Debian Lenny.

Třetí část se zabývá samotnou prezentací. Prezentace je provedena formou webových stránek. Webové stánky jsou založeny na redakčním systému Joomla, obsahují krátké animace ve flashi, které jednoduše znázorňují samotné chování traffic shapingu.

Práce obsahuje popisy, schémata strukturované sítě, ukázky, vysvětlení postupů a jednoduchých skriptů pro linux.

Přínosem této bakalářské práce je, že studenti provádějící měření v laboratořích najdou všechny potřebné informace na jednom místě.

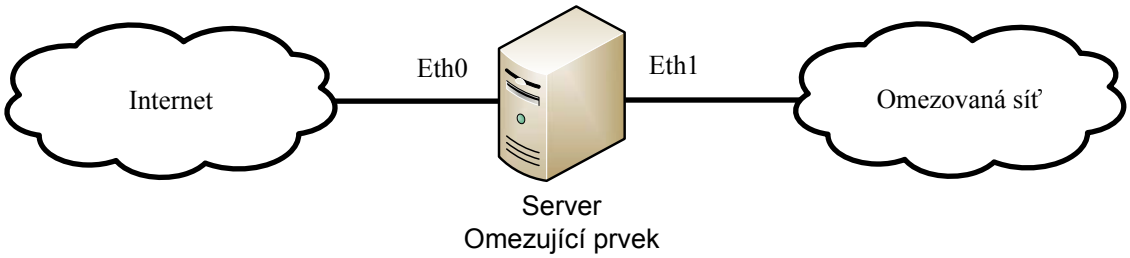
2 Teoretický rozbor problematiky traffic shaping

Omezování provozu lze rozdělit na dvě základní metody a to na queing a shaping.

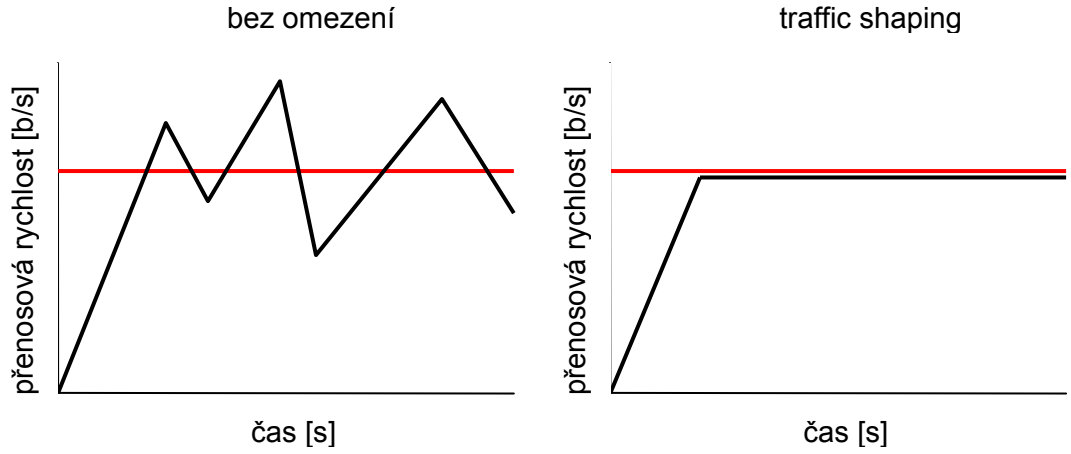
Queing – pakety řadí do fronty a zahazuje je jen při přetečení bufferu.

Pomocí shapingu propustíme jen určitý počet paketů a pakety přes limit zahodíme.

Je třeba si uvědomit, že neovlivníme jaká data a počet těchto dat, který nám z internetu, nebo ze sítě dojde. TCP protokol byl navržen pro nejefektivnější přenos. Když například web server začne posílat data směrem ke klientovi tak je posílá největší dosažitelnou rychlostí. Určitá část paketu se cestou ke koncovému počítači ztratí, nebo poškodí a je potřeba je pro úspěšné doručení poslat znovu. Serveru nedojdou zpět potvrzovací pakety tak sníží rychlost odesílání dat klientovi. Sníží tedy velikost využívaného pásma a tím ušetří i pásmo pro znovu odesílání nedoručených paketů. Tuto vlastnost protokolu využijeme tak, že budeme propouštět jen určitý počet paketů a zbytek zahodíme a tím docílíme snížení přenosové rychlosti, na kterou potřebujeme. Tento omezovací prostředek lze aplikovat pouze na pakety TCP, protože zahazování paketů jiných protokolů (UDP, ICMP, atd..) nezmění rychlost odesílaní. Efektivně ovládat můžeme pouze odchozí provoz, a pokud chceme ovládat jak odesílaní a přijímání tak data musí protéct dvěma rozhraními (obrázek 1). Graf přenosových rychlostí je na obrázku 2.



Obr. 1 Omezování dat oběma směry

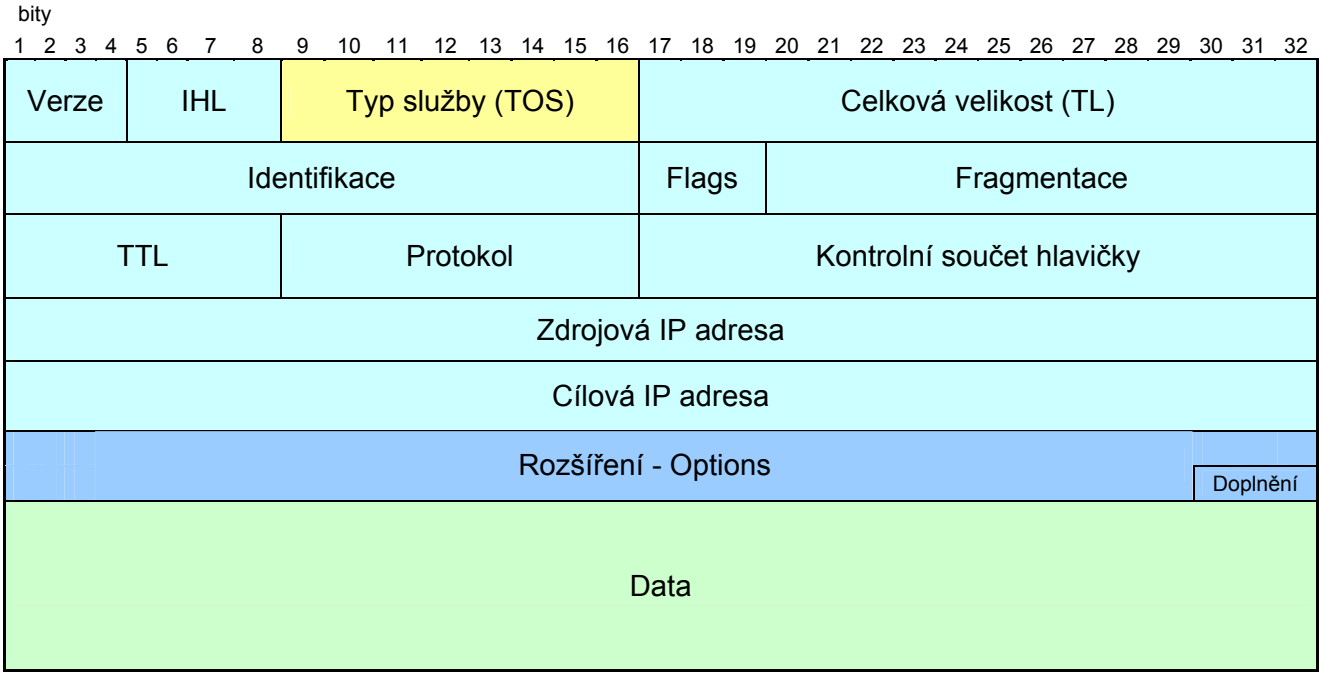


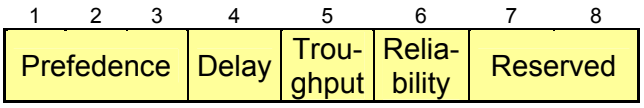
Obr. 2 Přenosové rychlosti

2.1 Důležité pojmy

- **Volná kapacita sítě** - rozdíl mezi celkovou dostupnou kapacitou a právě používanou kapacitou propustnosti sítě. Tato využitá kapacita se mění v čase, jiné využití je dopoledne, odpoledne a v nočních hodinách.
- **Ztrátovost paketů** - kolik procent paketů nedorazí od odesílatele k příjemci. Ke ztrátě může dojít v důsledku dočasného přetížení některého komponentu komunikační trasy, například po vyčerpání kapacity vyrovnávacích pamětí, přetížení procesoru routeru a podobně. Pro chování aplikací je rovněž důležitá i dynamika ztrátovosti, to je, zda se ztráty vyskytují ve shlucích.
- **Zpoždění** - doba odezvy. Příkaz ping využívá zprávy „Echo Request“ (výzva) a „Echo Reply“ (odpověď) protokolu ICMP. Doba odezvy se udává v ms.
- **Změna zpoždění** - jak se mění zpoždění jednotlivých paketů během přenosu. Tato vlastnost je pro aplikace často důležitější než absolutní, nebo průměrná doba zpoždění.
- **Qdisc** - Queueing Discipline. Algoritmus zajišťující rozdělování paketů do front a jejich následné vybírání.
- **Class** - jeden qdisc může být rozdělen do několika tříd (class). Ty se pak mohou dále dělit.

2.2 Typ služby TOS - Type of services





Tabulka 1: Paket IP v4

Pojmenování pole	Velikost	Popis
Prefedence	3 bity	Přednost: pole určuje prioritu paketu, 9 typů priorit. Od nejnižší po nejvyšší
Delay	1 bit	Zpoždění: Nastaveno na 0 na dotaz "normální" zpoždění při dodání; nastaveno na hodnotu 1 v případě, pokud je požadováno nízké zpoždění.
Troughput	1 bit	Propustnost: Nastavte na 0 na dotaz"normální" propustnost; nastaven na hodnotu 1, pokud požadujete vyšší propustnost.
Reliability	1 bit	Spolehlivost: Nastavte na 0 na dotaz "normální" spolehlivost; nastavena na 1, pokud požadujete vyšší spolehlivost doručení.
Reserved	2 bity	Rezervované, nepoužívají se.

Tabulka 2: Typ služeb (TOS)

Hodnota přednosti	Úroveň priority
000	normální
001	přednostní
010	okamžitá
011	blesková
100	zvýšená blesková
101	kritická
110	řízení propojení sítí
111	řízení sítí

Tabulka 3: Pole předností v TOS

Pole TOS se nachází v IP paketu (Tabulka 1) obsahuje 8 bitů (Tabulka 2). Používá se k vyjádření priority ve všech IP paketech. Tato hodnota umožňuje řízení kvality služeb (QoS). Tento mechanizmus je aplikován u IP paketů se zvýšenou prioritou, jakou je například přenos dat v IP telefonii. Router zpracovává IP pakety podle konfigurace a rozhoduje se, které IP pakety odešle první jen podle TOS pole. Nejdůležitější částí v TOS (Tabulka 3) je pole přednosti 3 bity. Pomocí 3 bitů lze docelit na 8 hladin. TOS dnes nahradilo modernější DSCP.

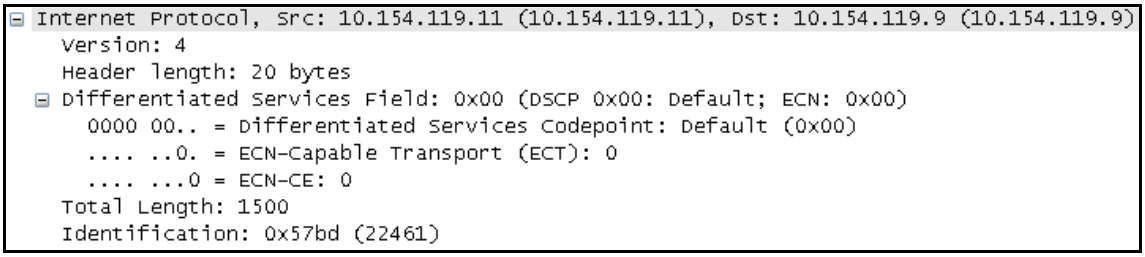
2.3 DSCP

DSCP je 6 bitová hodnota (2^6 a proto může nabývat 0-63 hodnot) části DS pole což je dnešní název pro bývalý TOS pole v IP v4 paketu a využívá se pro určení PHB - metody pro forwarding a fronty. DS pole je oktet v IP hlavičce třídy provozu, pokud je interpretován podle definice RFC2427. Při použití horních 3 bitů DSCP zpětně kompatibilní s IP precedence. Zbývající bity nejsou v současné době využity.

DSCP kódování má tři rozsahy:

- xxxxx0 - standardní akce
- xxxx11 - experimentální užití
- xxxx01 - EXP/LU - budoucí využití

Defaultně je DSCP 000000 toto jde vidět na zachyceném IP paketu ve wiresharku na obrázku 3.



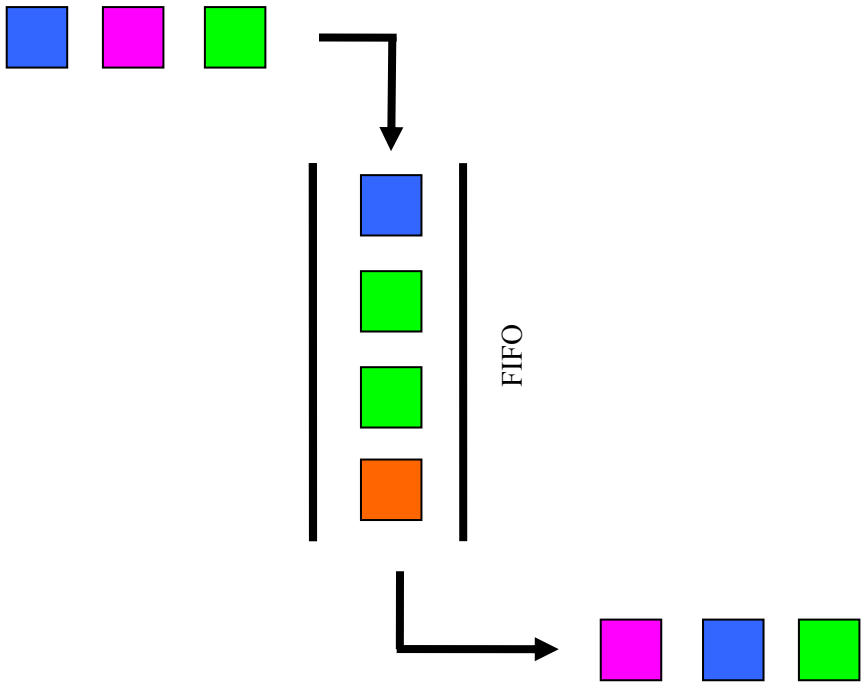
Obr. 3 DSCP zachycené ve Wiresharku

2.4 Beztřídní metody dělení

Classless Queueing Disciplines (qdiscs) Tyto metody jsou určené k zavěšení na listy stromu tříd, nebo přímo na síťové zařízení. Nelze na ně již navěsit další třídy nebo metody.

Každá z těchto řadících disciplín může být použita jako primární qdisc na rozhraní, nebo může být použita uvnitř jako list třídní metody qdisc. Toto jsou základní organizační pravidla použité pod Linuxem.

2.4.1 FIFO - First-in First-Out



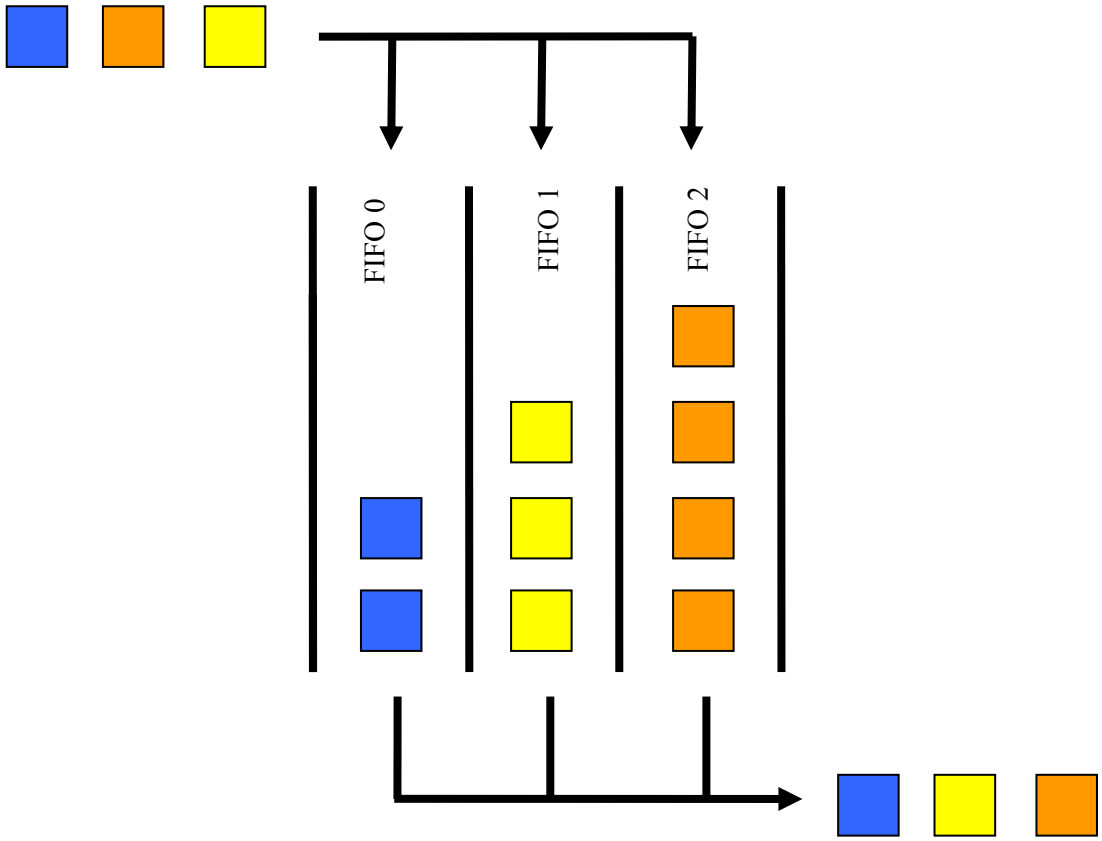
Obr. 4: FIFO

FIFO algoritmus vytváří standardní qdisc na všech linuxových síťových rozhraních (PFIFO Fast). Vykonává tvarování, nebo přeskupení paketů. Jednoduše přenáší pakety tak, jak rychle může po přijmutí a zařadí je do fronty. Toto také používá qdisc uvnitř všech nově vytvořených tříd dokud další qdisc, nebo třída nenahradí FIFO.

Reálný quedisc je však omezen velikostí bufferu a to proto, aby předcházel jeho přetečení v případech, když nemůže vyřazovat z fronty tak rychle jak je přijímá. Linux implementuje dva základní typy qdiscu, jeden je založen na bytech a druhý na paketech. Bez ohledu jaký typ FIFO používáme, velikost fronty je definována jako konstantní. Pro PFIFO jednotky rozumíme pakety a pro BFIFO jimi rozumíme byty.

2.4.2 P-FIFO Fast queuing discipline

Pakety se řadí do front ve skupinách založených na TOS a priorizaci.



Z fronty se vyřazují postupně od skupiny FIFO 0, poté FIFO 1 a nakonec FIFO 2

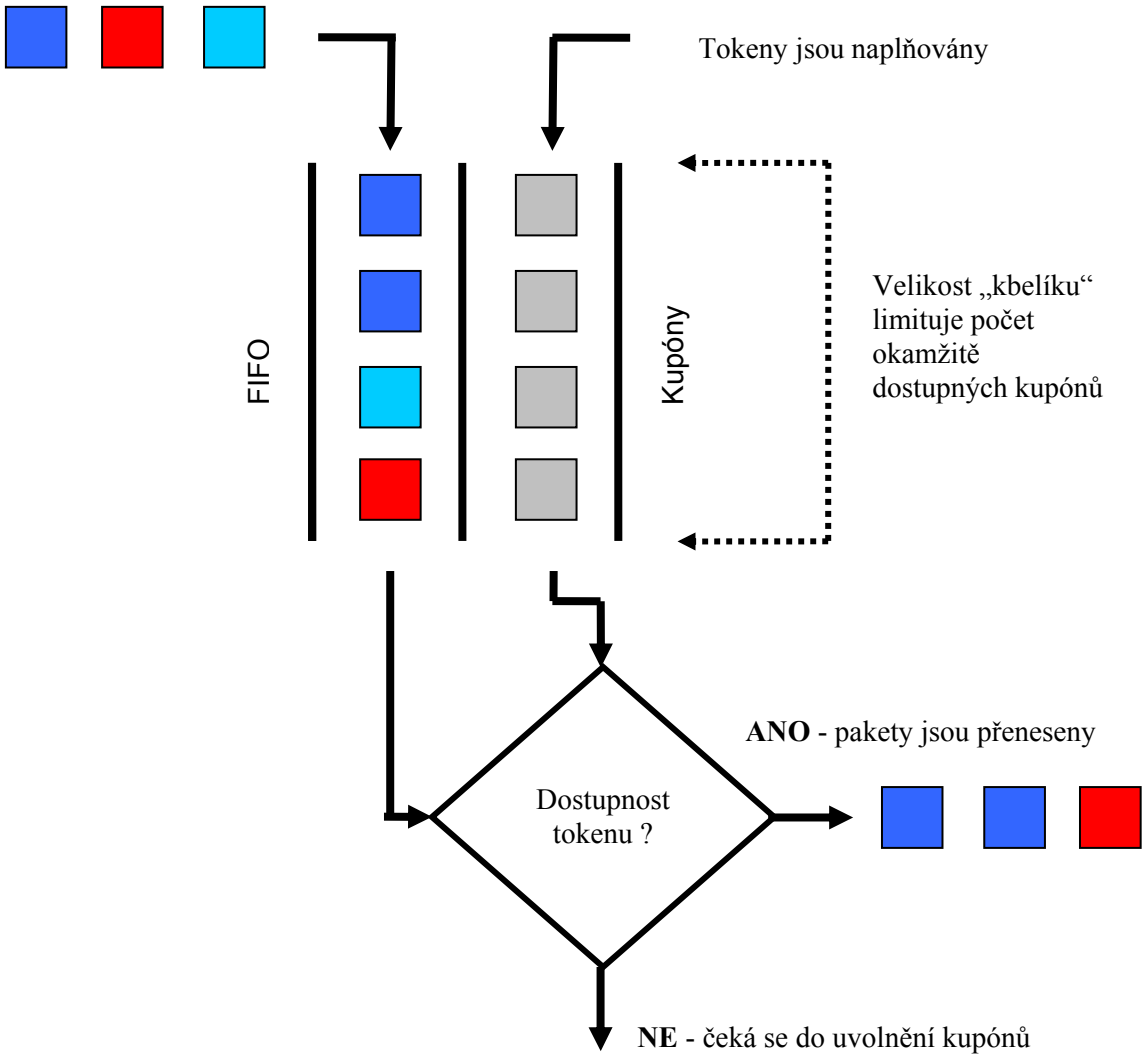
Obr. 5: P-FIFO

PFIFO-Fast qdisc je základní qdisc pro všechny rozhranní na Linuxu. Je založen na stanoveném FIFO qdiscu, tento qdisc provádí „určitou“ priorizaci. Vytváří tři různé skupiny (individuální FIFO) pro třídění provozu. Provoz s nejvyšší prioritou je umístěn ve skupině 0 a ta je vždy první vyprazdňována. Obdobně je na tom skupina 1, ta je vždy vyprázdněna z fronty paketů před poslední 2 skupinou.

V PFIFO qdisc se nedá nic nastavovat pro koncové uživatele. Pro přesné informace o prioritách nahlédněte do TOS, které se nachází v paketu IPv4. TOS je v tabulce 1 a tabulce 2.

2.4.3 TBF

Token bucket filter



Obr. 6: TBF

Tento qdisc je postaven na znacích a „kbelících“. Jednoduše tvaruje provoz vysílaných dat na rozhraní. Omezuje rychlost, se kterou vyřazuje pakety z fronty na jednotlivých rozhraních. Snižuje vysílání dat na určitou rychlost. Pakety jsou přeneseny, jestliže je dostupný dostatek znaků. Jinak jsou pakety zahozeny. Zpoždění paketů v tomto modelu představuje uměle zvýšenou odezvu způsobenou cestou paketu.

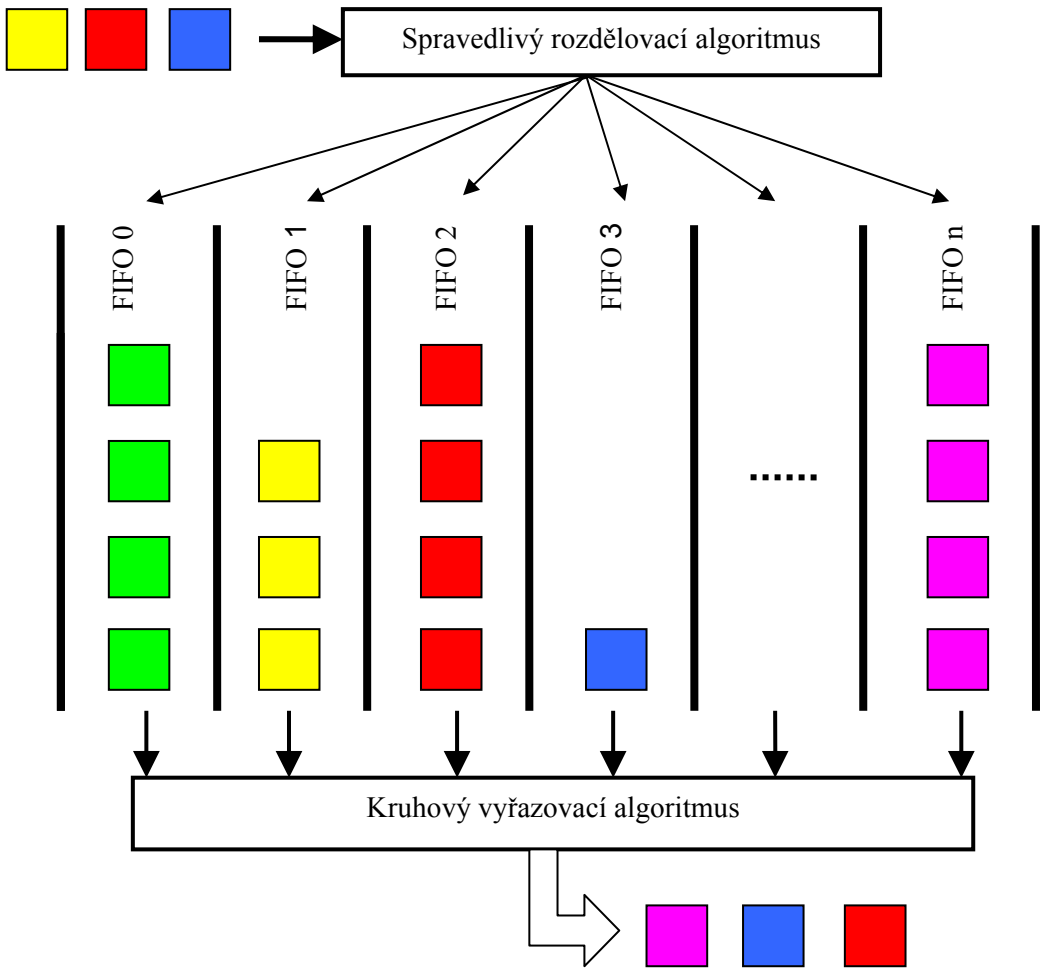
Algoritmus Token Bucket se popisuje pomocí následujícího přirovnání. Máme kbelík (bucket), do kterého se přidávají kupóny (tokens) zadanou rychlostí (burst rate nebo average traffic rate nebo CIR). Kbelík má určitou velikost (burst size Bc), když je plný,

tak se příchozí kupóny zahazují. Když přijde paket, tak se odeberou kupóny z kbelíku (1 kupón pro 1 byte paketu) a provede se conform-action (většinou se odešlou data). Pokud v kbelíku není dostatek kupónů, tak se na paket aplikuje exceed-action (většinou zahození).

Z popisu plyne, že tento algoritmus umožní přenést shluky dat o maximální velikosti Bc, ale záleží na předchozí komunikaci. Stejně tak je ovlivněna následující komunikace na průměrnou rychlost CIR. Jinak řečeno, můžeme přenášet data průměrnou rychlostí CIR a nárazově přenést Bc bytů navíc.

2.4.4 SFQ

Stochastic fair queuing



Obr. 7: SQF

SFQ qdisc se pokouší spravedlivě rozdělovat příležitost přenášení dat do sítě mezi libovolný počet toků. Splňuje použití hash funkce k rozdělení provozu mezi oddělené (vnitřně udržované) FIFO s vyřazení z fronty vybrané v kruhovém modelu. Protože se může stát, že se nespravedlivě vybere hash funkce, tak se tato funkce mění v pravidelných intervalech. Tímto se tato nespravedlivost eliminuje.

2.4.5 ESFQ

Extended SFQ

Rozšíření SFQ spočívá pouze v možnosti nastavovat parametry klasického SFQ.

2.4.6 WRR

Weightered Round Robin

Tato metoda rozděluje pakety do několika front. Každá z front má přiřazenou vlastní váhu - kvantum. Metoda cyklicky odesílá pakety po jednotlivých třídách, přičemž z každé třídy odešle maximálně kvantum bytů.

Používá se u starších switchů Cisco (například Catalyst 3550) se používal plánovač front (Queue Scheduler) WRR - Weighted Round Robin. Funguje jednoduše, Round Robin znamená, že se z výstupních front (na C3550 jsou 4, jedna může být nastavena na prioritní, ta se pak nepoužije) odebírají pakety postupně z každé fronty (kruhově 1, 2, 3, 4, 1, 2, atd.). Tím, že přidáme váhu jednotlivým frontám, zajistíme, že se z každé fronty odebere proporcionálně určitý počet paketů každé kolo. Tedy, že některá fronta je odbavována rychleji. Tím můžeme dosáhnout rozdělení pásma na portu pro určité fronty. Zařazování dat do front můžeme řídit.

2.5 Třídní metody

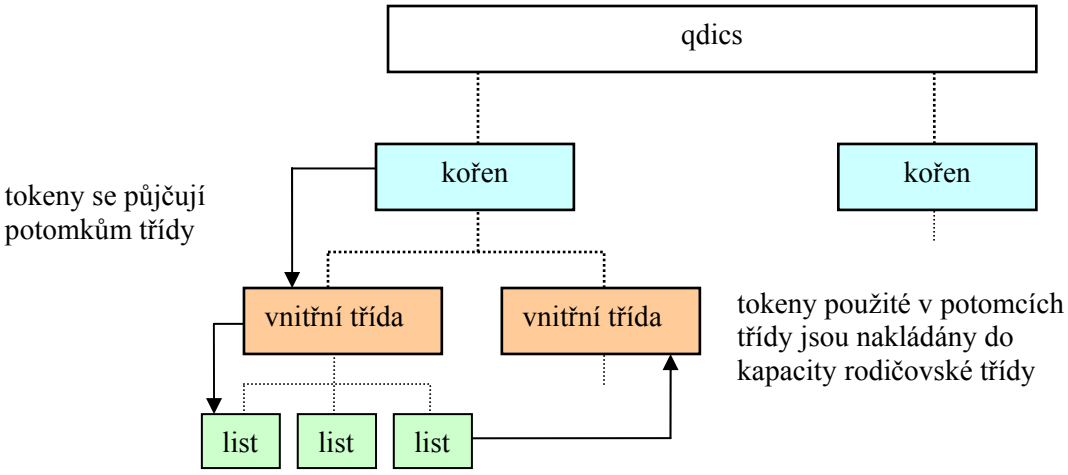
Classful Queuing Disciplines (qdiscs)

Kontrolu provozu pod Linuxem můžeme uvolnit přes agendu classful disc. Nesmíme zapomenout, že classful frontě mohou filtry k nim přiřazené, povolovat paketům přesměrování do konkrétní třídy a podtřídy.

Existuje několik běžných termínů k popisu tříd přímo připojených do kořene qdiscu a konečné třídy. Třídy připojené do kořene qdiscu nazýváme jako kořenové třídy a obecně jako vnitřní třídy. Jakémukoliv konci třídy v jednotlivé řadící disciplíně je znám jako list třídy. Analogie stromové struktury tříd.

2.5.1 HTB

Hierarchical Token Bucket



Obr. 8: Struktura tříd a zápůjček HTB

HTB je shaper s podporou tříd. Díky němu můžeme vytvořit stromovou strukturu - tj. pro každý počítač, nebo službu použijeme list stromu. Na tyto listy můžeme zavěsit některou z předešlých front, pokud žádnou frontu nezavěsíme, tak se použije standardní fronta FIFO.

Od českého autora Martin Devera pochází metoda qdisc HTB, která je méně složitá než CBQ, ale v mnoha případech plně dostačující. Podporuje podobné možnosti, lze shapovat i použít priority. Hodí se zvláště do situací, kdy je třeba pevnou šířku pásma rozdělit mezi více aplikací, uživatelů a služeb. Tím má každý zaručen garantovanou

šířku pásma a může využít vetší šířku pokud je právě dostupná. HTB poskytuje podobnou funkčnost jako CBQ, ale nepoužívá propočty nečinnosti linky. Lze spíše přirovnat k TBF s třídami. HTB je dostupné pouze na platformě Linux, kde je vyvinut jako náhrada CBQ se snazší konfigurací.

Zachovává vlastnosti CBQ pro které přidává konfiguraci rate a ceil. Rate je garantovanou šířkou pásma a ceil nastavuje maximální rychlost.

2.5.2 HFSC

Hierarchical Fair Service Curve

HFSC je první algoritmus, který se snaží obsáhnout tři důležité služby: garanci realtime provozu, adaptivní metodu besteffort a hierarchické sdílení linky. Jedná se o pokračovatele algoritmu HPFQ. V současnosti se jedná o nejmodernější shapovací algoritmus, který poskytuje podporu priority provozu a v návaznosti dokáže přerozdělovat tok linky realtimově podle aktuálního vytížení.

2.5.3 PRIO

Priority scheduler
PRIO je podobné `pfifo_fast`, při vytvoření vytvoří 3 třídy, které mají prioriy 0, 1 a 2 a zpracovávají se postupně (od nejnižší, teprve až je prázdná, může se zpracovat další třída). Jednotlivé třídy zpracovávají pakety úplně obyčejnou frontou, ale lze na ně pověsit libovolnou jinou qdisc.

2.5.4 CBQ

Class Based Queuing

CBQ je dlouho používaná a na nastavení velmi bohatá qdisc. Podporuje libovolné vnořování tříd a shaping. Třídy mohou půjčovat svou konektivitu podtřídám a získávat od svých předků. Podporuje také priority (obdobně jako PRIO).

Při výpočtech vychází z doby nečinnosti linky, průměrovaných velikostí paketů a šířky pásma. Mezi jednotlivými třídami pak vybírá algoritmem Wighted Round Robin.

2.6 Důvody nasazení omezování provozu

Důvodem výstavby je efektivní využití zakoupené internetové konektivity, nebo přístupu do sítě. Ideální stav je takové využití sítě, aby nedocházelo k přetěžování sítě a byla zajištěna kvalita služeb. Špatným nastavením tříd a priorit může dojít k tomu, že danou linku zbytečně nevyužijeme, nebo přetížíme.

Vytvořením tříd a správným nastavením lze zajistit kvalitnější přístup k internetu – síti a lepšímu přístupu k informacím.

Kvalita spojení a to hlavně odezvy a ztrátovost se nejvíc projevuje u bezdrátových sítí. Jakékoliv její přetížení má za následek velmi významné zvýšení odezvy.

2.7 Výhody použití

Pro použití jakéhokoliv systému musí klady převyšovat nad zápory.

- **Cena** – traffic shaping lze zprovoznit na operačním sytému Linux, který je volně ke stažení a zdarma.
- **Hardware** – pokud využíváme v síti nějaký server pro webovou prezentaci, nebo pro FTP atd. a je dostatečně výkonný pro nasazení shapingu je vhodné jej zprovoznit rovnou na něm. Tím ušetříme na hardwaru, na energiích a vyřeší to problém se **zodpovědností** a umístěním.
- **Snížení nákladů** – správným nastavením dojde k rozdělení internetové konektivity mezi uživatele.
- **Snížení zatížení sítě**
- **Tarifikace** – každé koncové zařízení může mít nastaveny jiné rychlosti pro odesílání a stahování dat.

2.8 Problémy použití

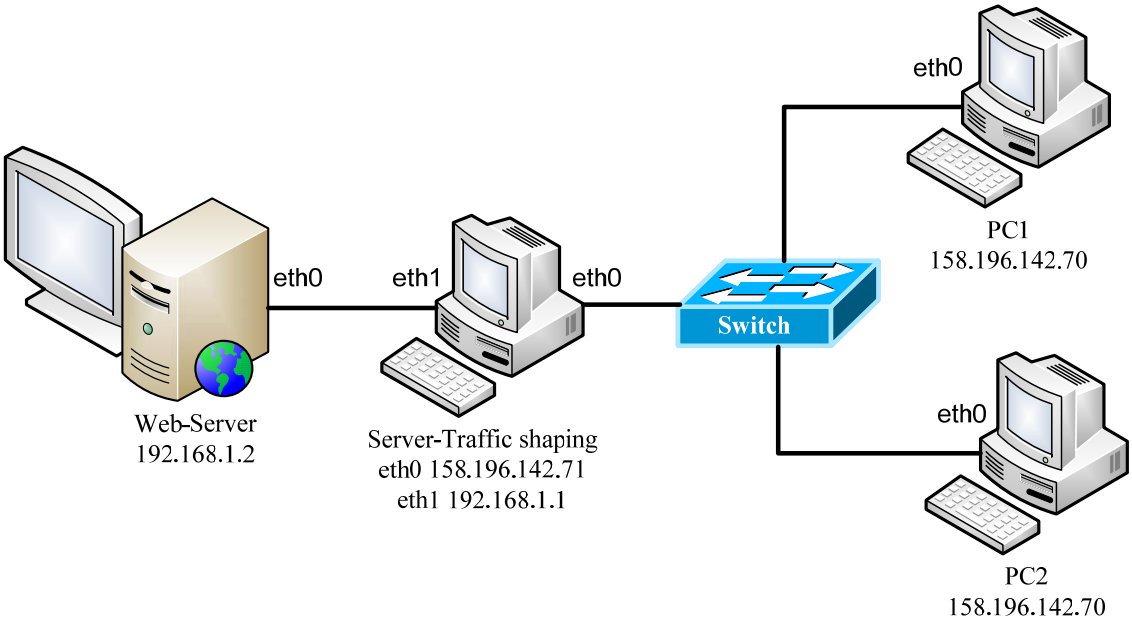
Každý systém má i své nevýhody, proto je nutné je zvážit a zařídit se tak, aby nás co nejméně omezovaly.

- **Hardware** – je potřeba dbát zvýšené opatrnosti při volbě hardwaru, na kterém systém poběží. Po instalaci je vhodné systém nechat zahořet na méně důležité části sítě a vyzkoušet v praxi jeho správnou funkčnost. Tím se odstraní případné chyby před ostrým provozem. Každé zařízení nedokáže pracovat 24 hodin denně a 365 dní v roce po několik let. Z celého systému nejvíc trpí pevné disky, které se při nesprávném chlazení přehřívají a vykazují častější a častější chyby.
- **Problémy s NAT** – pokud proběhne překlad IP adresy tak už není možné určit její IP adresu a tím nemůžeme filtrovat a řadit do skupit podle IP adres.
- **Zodpovědnost a role** – ve společnosti je třeba vyřešit, kdo se bude o systém starat a udržovat jej stále aktuální a v provozu.
- **Uplatnění změn** – při změně v sítí, navýšení kapacity internetové konektivity, zvýšení propustnosti mezi prvky v síti se musí provézt úpravy v konfiguraci. Pro pár prvků v síti je úprava skriptu otázkou několika minut, ale pro rozsáhlou síť se stovkami koncových zařízení je to nemožné. Pro tyto případy jsou zde komerčně prodávané systémy jako Netprovider, nebo ISPadmin.

3 Návrh laboratorního pracoviště

3.1 Zapojení počítačů

Zapojil jsem počítače podle obrázku 9. K propojení jsem použil přímé kabely a switch v racku. Koncových zařízení – počítačů můžeme zapojit i víc, jen je musíme připsat do později spouštěných skriptů. Pokud budeme provádět čistou instalaci OS Debian, nebo zkoušet testovat pod Vmware tak je postup instalace tohoto OS součástí přílohy 1.



Obr. 9: Schéma zapojení počítačů v laboratoři

3.2 Instalace balíčků

Pro správnou funkci potřebujeme jádro systému s podporou HTB a je tedy nutná verze 2.4.20 a výše, nebo použít patch na starší verzi jádra. K fungování shapovaní potřebujeme mít nainstalované dva balíčky a to: iptables a iproute.

Pro testování je potřeba mít nainstalován balíček apache2.

Verzi jádra zjistíme příkazem v příkazové řádce:

```
Server:/# uname -r
2.6.18-5-686
```

systém vypsal: 2.6.18-5-686

Co tato čísla znamenají?

2.6 je jádro (kernelu) a 18-5 je jeho verze a 686 značí instrukční sadu procesoru typu 686 – 686 je nástupce navazující na procesory 386.

Jestli máme nainstalované potřebné baličky, zjistíme velmi jednoduše:

```
Server-traffic-shaping:/# dpkg -l | grep iptables
ii  iptables 1.4.2-6 administration tools for packet filtering and NAT
Server:/# dpkg -l | grep iproute
ii  iproute  20080725-2 networking and traffic control tools

Server-traffic-shaping:/# dpkg -l | grep apache2
ii  apache2 2.2.11-2 Apache HTTP Server metapackage
```

Baličky jsou již součástí instalace a je tedy vše připraveno.

Chybějící balíčky doinstalujeme:

```
apt-get update
```

Tímto příkazem stáhneme nejnovější repositáře – seznamy balíčků

```
apt-get install apache2
```

Tímto příkazem nainstalujeme požadovaný balíček.

3.3 Napsáni skriptu

Samotný skript se skládá z několika částí:

- vymazání qdiscu
- vytvoříme HTB qdisc
- vytvoříme hlavní kořenovou třídu
- vytvoříme větve – podtřídy kořene
- na každou třídu „pověsíme“ SFQ
- označkujeme pakety
- provoz podle označkování roztřídíme pomocí filtru

3.3.1 Příkazy skriptu

Příkazem vymažeme všechny qdisc na síťovém rozhraní eth0

```
tc qdisc del dev eth0 root
```

Příkazem vytvoříme vlastní HTB qdisc na rozhraní eth0

```
tc qdisc add dev eth0 root handle 1:0 htb default 14
```

Tímto příkazem si vytvoříme kořenovou třídu pojmenovanou 1:0 s maximální propustností 1024 kbit (za sekundu).

```
tc class add dev eth0 parent 1:0 classid 1:1 htb rate 1024kbit
```

Z této třídy se větví další čtyři třídy, každá s jinou garantovanou rychlostí.

```
tc class add dev eth0 parent 1:1 classid 1:11 htb rate 512kbit ceil 1024kbit
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 256kbit ceil 512kbit
tc class add dev eth0 parent 1:1 classid 1:13 htb rate 64kbit ceil 256kbit
```

Na každou koncovou třídu pověsíme jednu z ne-classful disciplínu SFQ. Pokud neuvedeme nic, předpokládá se, že chceme připojit standardní frontu FIFO. Přepočítávání spojení proběhne každou sekundu, proto ta jednička na konci.

```
tc qdisc add dev eth0 parent 1:11 handle 11:0 sfq perturb 1
tc qdisc add dev eth0 parent 1:12 handle 12:0 sfq perturb 1
tc qdisc add dev eth0 parent 1:14 handle 13:0 sfq perturb 1
```

Aby nám filtr pakety přiřadil do správné třídy je potřeba pakety „označkovat“. Všechny pakety označkujeme značkou 99, aby neprošel žádny neoznačkovaný paket a potom označkujeme zbytek podle cílové adresy. Značka 99 je tedy "defaultní" a je nastavená všem parketům, které se neoznačkují jinými pravidly.

```
iptables -t mangle -A POSTROUTING -j MARK --set-mark 99
iptables -t mangle -A POSTROUTING -d 158.196.142.70 -j MARK --set-mark 1
iptables -t mangle -A POSTROUTING -d 158.196.142.70 -j MARK --set-mark 2
```

Zde filtr pozná podle značky, do jaké třídy má paket zařadit.

```
tc filter add dev eth0 parent 1:0 protocol ip handle 1 fw flowid 1:11
tc filter add dev eth0 parent 1:0 protocol ip handle 2 fw flowid 1:12
tc filter add dev eth0 parent 1:0 protocol ip handle 99 fw flowid 1:13
```

3.3.2 Tvorba a spouštění skriptů

Skripty pro spouštění a vypínání shapování můžeme napsat několika způsoby. Jednodušší způsob je vytvořit si několik textových souborů a ty spouštět. Složitější, ale elegantnější řešení je napsat jeden komplexnější skript. Soubory vytvořím v jedné složce.

Složku vytvoříme příkazem:

```
mkdir shaping
```

3.3.3 Jednodušší řešení:

```
nano shaping-start
```

příkaz vytvoří soubor a do něj vypíšeme postupně všechny příkazy až na první příkaz pro vymazání qdiscu – ten by nám házel chybovou hlášku při prvním spuštění. Soubor uložíme klávesou zkratkou Ctrl+X a potvrzením y (yes), nebo a (ano)

```
nano shaping-stop
```

do tohoto souboru vložíme jen příkaz pro smazání qdiscu

```
tc qdisc del dev eth0 root
```

a uložíme

```
nano shaping-info
```

do tohoto souboru vložíme příkazy:

```
tc filter show dev eth0
tc -d -s qdisc show dev eth0
tc -d -s class show dev eth0
```

a uložíme.

Abychom mohli skripty spustit, je třeba jim změnit práva a to příkazem:

```
chmod 777 shaping-start
```

Skript spustíme příkazem:

```
./shaping-start
```

V konzoli to vypadá přesně takto:

```
Server:/shaping#./shaping-start
```

Teď si můžeme nechat vypsat informace o třídách stromu příkazem:

```
./shaping-info
```

Shaping vypneme příkazem:

```
./shaping-stop
```

3.3.4 Složitější řešení

Obsah skriptu:

```
start (){
#zde vložíme obsah skriptu shaping-start
    }
stop (){
#zde vložíme obsah skriptu shaping-stop
    }

info (){
#zde vložíme obsah skriptu shaping-info
      }
case "$1" in
       start)
              start
              ;;
       stop)
              stop
              ;;
       info)
              info
              ;;
       *)
              echo "Usage shaping{start|stop|info}"
              ;;
esac
exit 0
```

Soubor uložíme pod názvem shaping. Skript poté spouštíme příkazy:

```
./shaping start
./shaping info
./shaping stop
```

3.4 Praktická realizace

3.4.1 Nastavení web serveru

Nastavíme IP adresy a defaultní routu:

```
ifconfig eth0 192.168.1.2 netmask 255.255.255.0
route add default gw 192.168.1.1
```

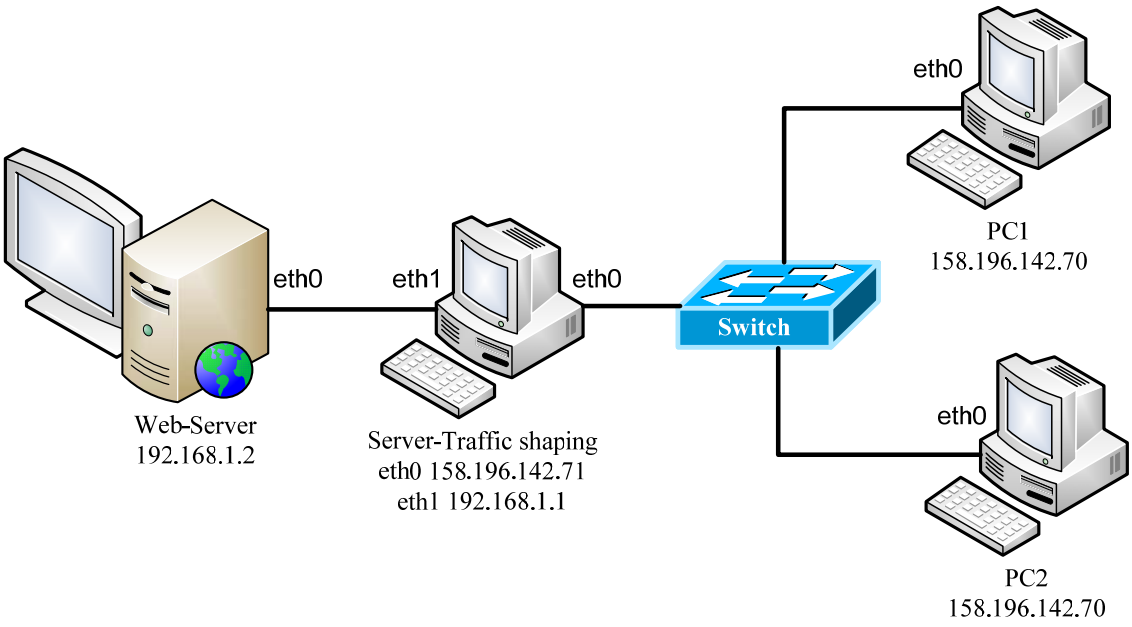
Nastavení webhostingu po umístění testovacího souboru:
V konfiguračním souboru web serveru `/etc/apache2/sites-enabled/000-default` zakomentujeme #, nebo smažeme sedmnáctý řádek:

```
RedirectMatch ^/$ /apache2-default/
```

Tak už se nám nebude web přesměrovávat na složku apache2-default, která slouží k testovacím účelům, a rovnou se dostaneme do rootu webu.

Ted stačí překopírovat soubor, který budeme stahovat do složky s webem.

Já jsem si stáhnul cca 110 MB velký soubor s instalací openoffice a přejmenoval ho na soubor test (bez přípony). Tento soubor jsem zkopíroval do /var/www/



Obr. 9: Schéma zapojení počítačů v laboratoři

3.4.2 Nastavení traffic shaping serveru

Zde povolíme routování, neboli předávání paketů z jednoho rozhraní na druhé a to příkazem:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Na druhé sítové kartě nastavíme ip adresu pro spojení s webovým serverem příkazem:

```
ifconfig eth1 192.168.1.1 netmask 255.255.255.0
```

Pro omezování provozu upravíme a spustíme skripty. Pokud bychom chtěli omezovat i odesílaná data stačí skript zkopírovat a upravit v něm vytvoření vlastního HTB qdiscu na druhé rozhraní a to na eth1. Tímto budeme omezovat i odchozí datový provoz.

3.4.3 Natavení koncových PC

Na koncových počítačích nastavíme defaultní bránu příkazem:

```
route add defaut gw 158.196.142.71
```

3.4.4 Testování omezování provozu a jeho vliv na odezvu

Pro samotné testování jsem si nastavil přenosové rychlosti takto:

pro PC1 minimum 512kbit s maximálním stropem 1024kbit

pro PC2 minimum 512kbit s maximálním stropem 2048kbit

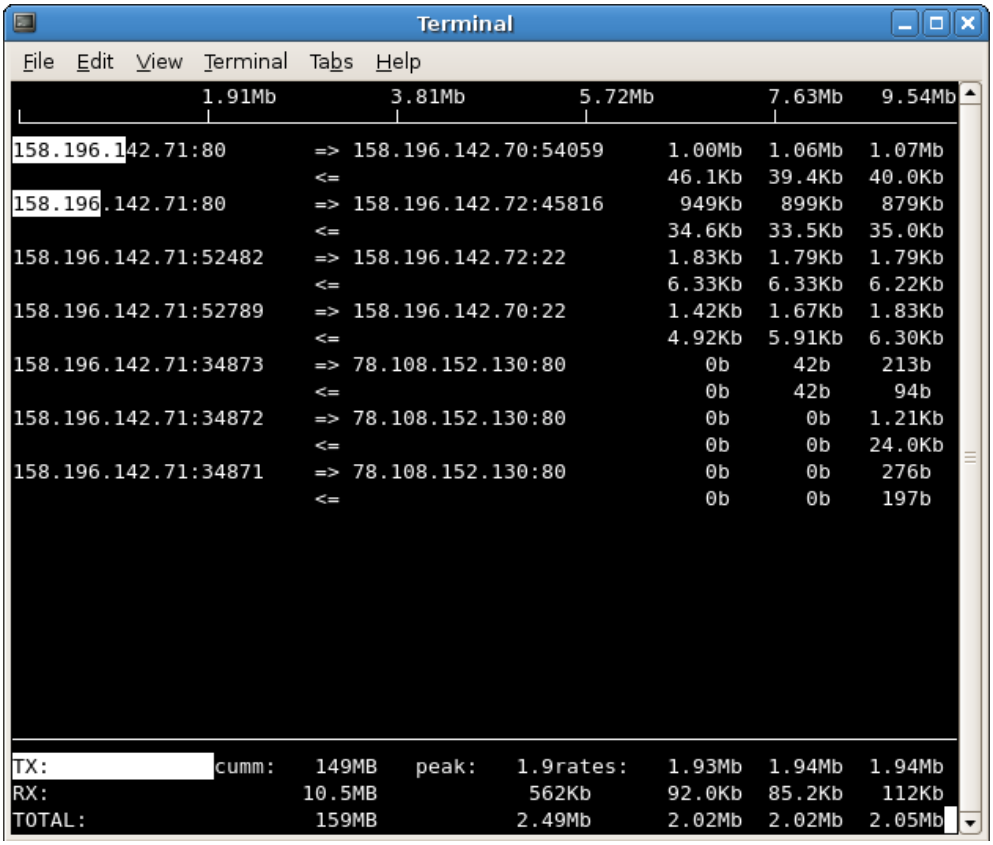
Spustil jsem skript a sledoval přenosové rychlosti a vliv na změnu odezvy. Odezvu jsem zjišťoval příkazem z PC1 a PC2:

```
ping 192.168.1.2
```

Přenosové rychlosti lze sledovat jak na straně web serveru tak na serveru traffic shapingu, já jsem je sledoval na serveru traffic shapingu a to příkazem:

```
iftop –i eth0
```

Odezvy a přenosové rychlosti jsem zkopíroval z konzole a graf přenosových rychlostí iftop zachytil jako snímek obrazovky. Průběh je zachycen na obrázku 10.



Obr. 10: Výpis iftop

Po spuštění shapingu je patrná změna odezvy na výpisu:

```
PC1:~# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.400 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.395 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.394 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.394 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=46.9 ms
64 bytes from 192.168.1.2: icmp_seq=6 ttl=64 time=34.8 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=46.8 ms
64 bytes from 192.168.1.2: icmp_seq=8 ttl=64 time=58.8 ms
64 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=3.04 ms
64 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=38.4 ms
64 bytes from 192.168.1.2: icmp_seq=11 ttl=64 time=26.7 ms
64 bytes from 192.168.1.2: icmp_seq=12 ttl=64 time=38.4 ms
64 bytes from 192.168.1.2: icmp_seq=13 ttl=64 time=26.6 ms
64 bytes from 192.168.1.2: icmp_seq=14 ttl=64 time=42.6 ms
64 bytes from 192.168.1.2: icmp_seq=15 ttl=64 time=30.5 ms
64 bytes from 192.168.1.2: icmp_seq=16 ttl=64 time=0.396 ms
64 bytes from 192.168.1.2: icmp_seq=17 ttl=64 time=0.395 ms
64 bytes from 192.168.1.2: icmp_seq=18 ttl=64 time=0.397 ms

--- 192.168.1.2 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 22060msrtt
min/avg/max/mdev = 0.394/21.176/58.810/19.747 ms
```

Přenosové rychlosti jsem testoval stahováním soboru test z webového serveru příkazem:

```
wget 192.168.1.2/test
```

Přenosové rychlosti zachycené z konzole – stahuje data pouze jedno PC

```
PC1:~# wget 192.168.1.2/ test
--10:15:20-- http://192.168.1.2/ test
 => `test'
Connecting to 192.168.1.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117,979,202 (113M) [text/plain]

 1% [ ] 1,752,536 129.28K/s ETA 17:01
pc4n312:~#

PC2:~# wget 192.168.1.2/ test
--11:17:46-- http://192.168.1.2/ test
 => `test'
Connecting to 192.168.1.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117,979,202 (113M) [text/plain]

 1% [ ] 1,701,088 239.87K/s ETA 08:33
pc6n312:~#
```

Přenosové rychlosti zachycené z konzole –stahují data najednou obě PC

Zde je patrné rozdělení rychlostí pro oba počítače.

```
PC1:~# wget 192.168.1.2/test
--10:27:47-- http://192.168.1.2/test
 => `test'
Connecting to 192.168.1.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117,979,202 (113M) [text/plain]

 0% [ ] 978,536 106.01K/s ETA 21:07
pc4n312:~#

PC2:~# wget 192.168.1.2/test
--11:27:28-- http://192.168.1.2/test
 => `test'
Connecting to 192.168.1.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 117,979,202 (113M) [text/plain]

 1% [ ] 1,194,288 133.27K/s ETA 16:17
```

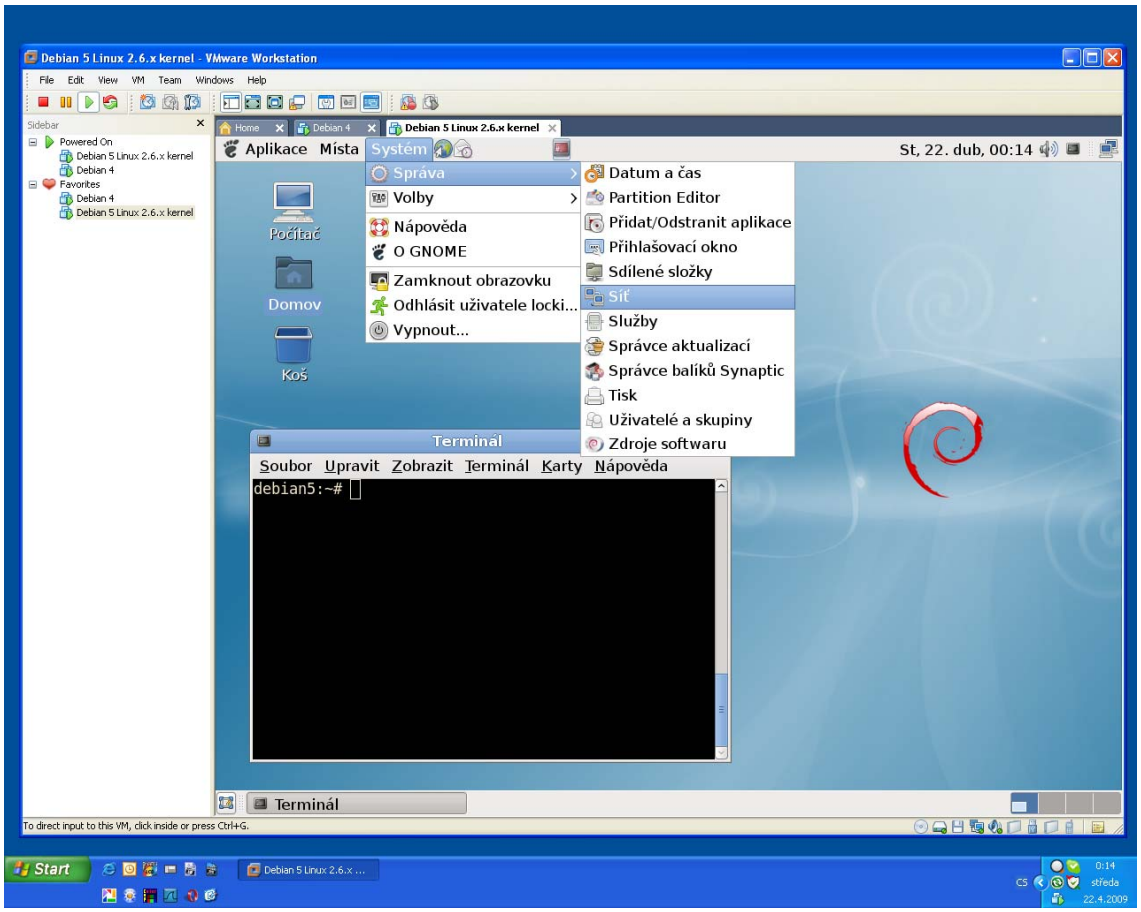
Změna odezvy při stahování obou počítačů najednou

```
PC1:~# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=9.58 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.486 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=26.9 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=22.8 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=34.5 ms
64 bytes from 192.168.1.2: icmp_seq=6 ttl=64 time=14.5 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=18.8 ms
64 bytes from 192.168.1.2: icmp_seq=8 ttl=64 time=10.7 ms
64 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=0.480 ms
64 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=22.6 ms
64 bytes from 192.168.1.2: icmp_seq=11 ttl=64 time=34.1 ms
64 bytes from 192.168.1.2: icmp_seq=12 ttl=64 time=26.6 ms
64 bytes from 192.168.1.2: icmp_seq=13 ttl=64 time=14.5 ms
64 bytes from 192.168.1.2: icmp_seq=14 ttl=64 time=6.52 ms
64 bytes from 192.168.1.2: icmp_seq=15 ttl=64 time=6.49 ms
64 bytes from 192.168.1.2: icmp_seq=16 ttl=64 time=46.1 ms
64 bytes from 192.168.1.2: icmp_seq=17 ttl=64 time=6.19 ms
64 bytes from 192.168.1.2: icmp_seq=18 ttl=64 time=14.3 ms

--- 192.168.1.2 ping statistics ---
18 packets transmitted, 18 received, 0% packet loss, time 17065ms
rtt min/avg/max/mdev = 0.480/17.600/46.187/12.260 ms


PC1:~# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=49.7 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=33.1 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=29.1 ms
64 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=16.9 ms
64 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=0.560 ms
64 bytes from 192.168.1.2: icmp_seq=6 ttl=64 time=8.66 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=32.5 ms
64 bytes from 192.168.1.2: icmp_seq=8 ttl=64 time=28.2 ms
64 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=40.8 ms
64 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=35.9 ms
64 bytes from 192.168.1.2: icmp_seq=11 ttl=64 time=32.1 ms
64 bytes from 192.168.1.2: icmp_seq=12 ttl=64 time=23.9 ms
64 bytes from 192.168.1.2: icmp_seq=13 ttl=64 time=19.8 ms
64 bytes from 192.168.1.2: icmp_seq=14 ttl=64 time=12.0 ms
64 bytes from 192.168.1.2: icmp_seq=15 ttl=64 time=7.60 ms
64 bytes from 192.168.1.2: icmp_seq=16 ttl=64 time=19.7 ms
64 bytes from 192.168.1.2: icmp_seq=17 ttl=64 time=35.3 ms

--- 192.168.1.2 ping statistics ---
17 packets transmitted, 17 received, 0% packet loss, time 16012ms
rtt min/avg/max/mdev = 0.560/25.084/49.733/12.708 ms
```



Obr. 11: Spuštěný Debian Lenny ve Vmware pod Windows XP

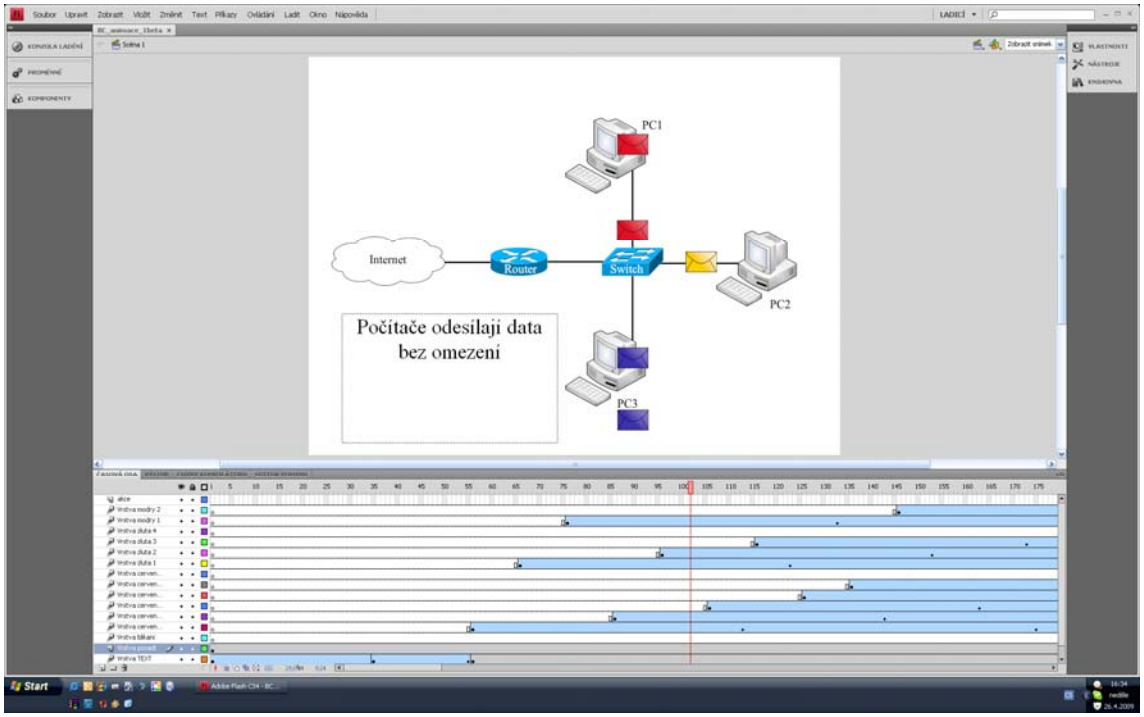
Pro testování funkčnosti jsem v domácích podmínkách používal program Vmware. Vmware je vizualizační nástroj, který umožňuje již na spuštěném OS vytvářet další virtuální počítače. V tomto nástroji jsem si vytvořil 3 virtuální počítače. Jeden systém s OS Debian Lenny jako server a dva systémy s OS Debian Etch jako stanice. Takto jsem mohl doma vyzkoušet chování na jediném počítači. Pro praktické testování jsem využil vybavení učebny N312 na naší katedře. Počítače zapojil dle schématu na obrázku 9.

4 Grafická prezentace

Pro tvorbu flash animací grafické prezentace jsem používal vývojové prostředí od Adobe a pro webovou prezentaci jsem se rozhodl použít redakční systém Joomla.

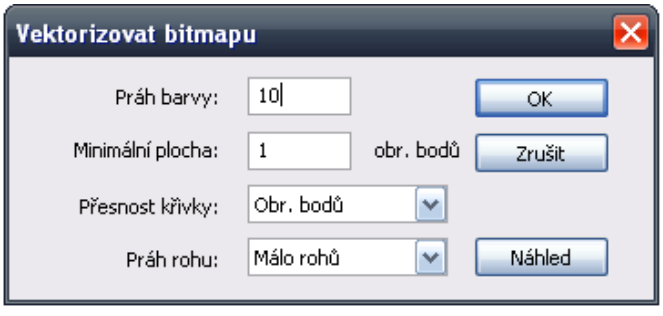
4.1 Flash animace

Animace jsem vytvářel v Adobe Flash CS4. Program pracuje s vrstvami, v každé vrstvě a v každém čase můžeme definovat pohyb, natočení, text atd..



Obr. 12: Tvorba animace v Adobe Flash CS4

Pro lepší práci s objekty je vhodné bimapovou grafiku převézt na vektorovou.
Změnit -> Bitmapa -> Vektorizovat Bitmapu (Obrázek 13)

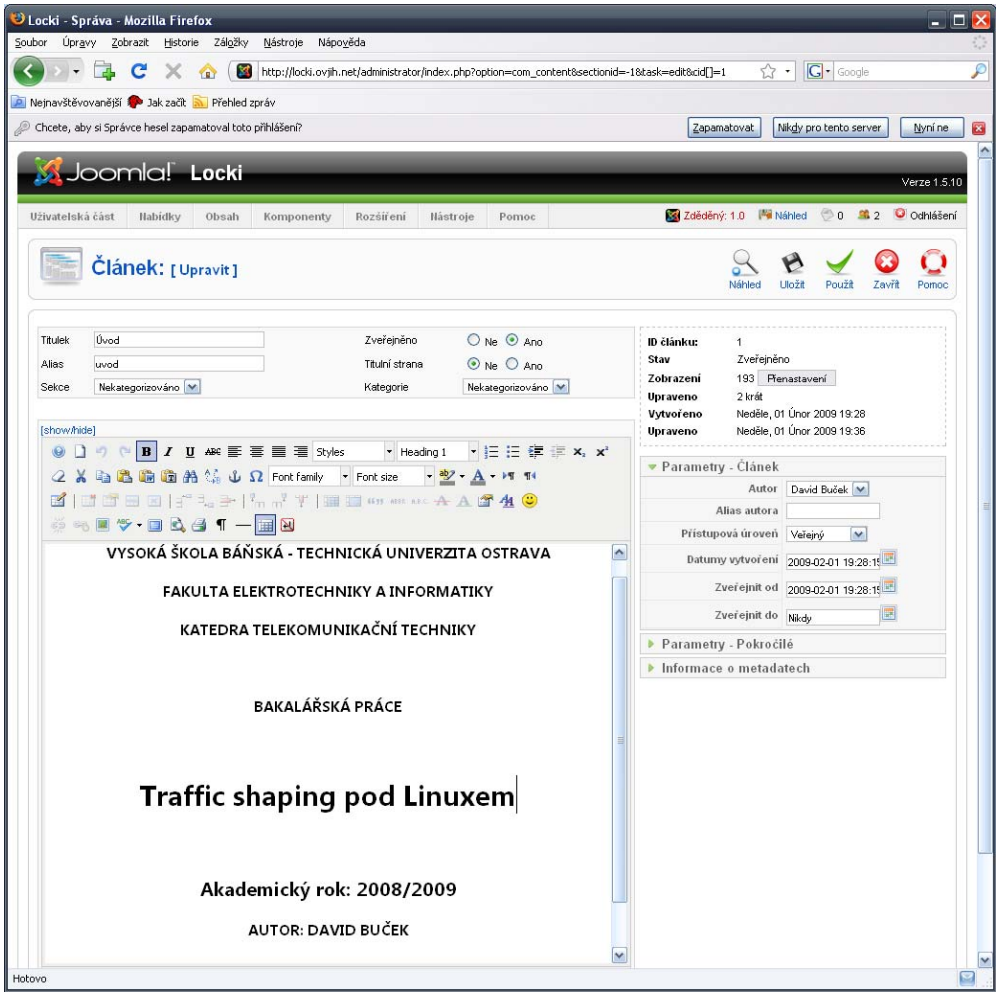


Obr. 13: Vektorizace bitmapy

Po převodu na vektorovou grafiku vypadají objekty mnohem lépe a netrpí JPEG kompresí. Vytvořená animace je ve formátu FLA a pro webovou prezentaci musí být ve formátu SWF. Na formát SWF převedeme animaci klávesou zkratkou Ctrl+Shift+Alt+S, nebo výběrem z menu: Soubor -> Exportovat -> Exportovat film. Výsledná animace je přehratelná ve všech internetových prohlížečích.

4.2 Webová prezentace

Pro webovou prezentaci jsem použil redakční systém Joomla 1.5. Pro tvorbu webu v tomto systému není třeba znát mnoho příkazů a tvorba obsahu je velmi podobná psaní dokumentu v MS Word. Postup instalace je popsán v příloze č. 1. Vytvořená web prezentace připravena k instalaci je příloha č. 4. Z webu je prezentace dostupná na http://buc102.ovjih.net , nebo na http://homel.vsb.cz/~buc102/ .



Obr. 14: Tvorba obsahu webu v Joomla 1.5

5 Závěr

V teoretické části práce jsem popsal klady, zápory a důvody nasazení traffic shapingu. Jsou zde taky popsány různé modely shapingu.

Hlavním cílem této práce bylo zprovoznit traffic shaping a vytvořit doprovodné animace. Řešení navržené v této bakalářské práci je plně funkční. Toto řešení je zjednodušeno pro použití ve výuce pro měření se studenty v laboratoři. Není nutná složitá kompilace jádra systému. Pro testování je použito metody HTB pro svou jednoduchost a velmi snadnou konfiguraci. Studenti, kteří budou provádět měření v předmětu práce v počítačových sítích, si mohou vyzkoušet víc konfigurací shapingu a vyhodnotit vliv na přenosové rychlosti a odezvy.

Všechny skripty, které jsem v práci použil, jsou vypáleny v textové podobě na přiloženém CD. Součástí tohoto CD je také zálohovaná webová prezentace tvořená v redakčním systému Joomla. Součástí CD jsou flash animace ve zdrojové i spustitelné formě.

Tato práce ukázala, že k vytvoření funkčního omezování provozu pomocí traffic shapingu není nutné vlastnit žádný speciální, nákladný hardware a k omezování provozu je ideální využít volně šiřitelný operační systém Linux.

Problematika traffic shapingu je velice obsáhlá a další rozšíření této práce by mohlo být vytvoření flash animací pro více metod shapingu, využití traffic shapingu na zařízeních firem Mikrotik, Cisco, nebo prakticky porovnat chování některých modelů ve větších sítích.

Literatura

[1] DEVERA, Martin. HTB Linux queuing discipline manual : user guide [online]. Dostupný z WWW: <http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm>.

[2] DEVAINE, Max. Traffic shaping : patchování a instalace [online]. 2006. Dostupný z WWW: <http://www.abclinuxu.cz/clanky/site/traffic-shaping-patchovani-a-instalace>.

[3] The TCP/IP Guide [online]. 2005. Dostupný z WWW: <http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm>.

[4] Cisco QoS : úvod do Quality of Service a DiffServ [online]. 18.1.2009. Dostupný z WWW: <http://www.samuraj-cz.com/clanek/cisco-qos-1-uvod-do-quality-of-service-a-diffserv/>.

[5] PODGORNÝ, Radek. HTB : úvod [online]. 2003. Dostupný z WWW: <http://www.root.cz/clanky/htb-jemny-uvod/>.

[6] Cisco QoS : omezování rychlosti - Policing, Shaping [online]. 2009. Dostupný z WWW: <http://www.samuraj-cz.com/clanek/cisco-qos-3-omezovani-rychlosti-policing-shaping/>.

[7] WEIS, Nate. Flash MX 2004 : pro vývojaře webových aplikací. Brno : Zoner Press, 2004. 528 s. ISBN 80-86815-12-9.

[8] Adobe Flash Action script 3 : button tutorial [online]. 2008. Dostupný z WWW: <http://www.tutvid.com/tutorials/flash/tutorials/as3AndButtons.php>.

Přílohy:

Seznam příloh na CD:

1. Popis instalace OS Debian Lenny
2. Popis instalace redakčního systému Joomla 1.5
3. Skripty v textových souborech
4. Instalační balíček kompletní webové prezentace
5. Animace ve formátu zdrojových FLA a přehratelných SWF
6. Instalační obraz ISO OS Debian Lenny